A Dynamic Trust Network for Autonomy-Oriented Partner Finding

Hongjun Qiu*

Abstract

The problem of partner finding is to identify which entities (agents) can provide requested services from a group of entities. It can be found in open and distributed environments for such tasks as file sharing and resource allocation. Previous studies have shown that entities can refine and determine partners through measuring trust relationships, i.e., the beliefs of entities that others will accomplish a request for assigned services at hand. Entities dynamically change their beliefs through recalling their past experiences to quickly identify partners for new requests. This paper aims to observe whether those changes can enable entities efficiently find partners and hence provide services. We propose a dynamic network, which contains trust relationships as links among entities (nodes). Then, we investigate its structure and efficiency in the moments with different trust relationships. Autonomy-Oriented Computing (AOC) is applied to observe how the network dynamics emerge from local behaviors. A notion of autonomy is embodied in defining how entities activate their partner finding behaviors, whereas self-organization is realized to update the strength of trust relationships. Experimental results explicitly display a dynamic process of this network, changing from containing no link to having some stable links. Specially, in this process, the efficiency gradually gets enhanced.

1 Introduction

In a multi-entity network, entities (e.g., companies) usually look for some others (i.e., finding partners) to provide services beyond their own limited abilities. The problem of partner finding appears widely in real-world applications. Here, we take cloud computing as an example. In cloud computing, vendors (e.g., Google, Amazon and Microsoft) deploy numerous servers over the Internet, to store data or run programs in real time as requested by users [3]. The servers work together to deal with such requests from hundreds or thousands of users per second [4]. Hence, it is quite important for servers to immediately identify who can work together to accomplish requests.

Currently, researchers have introduced a notion of trust relationship as a measurement for identifying partners, to solve real-world problems in open, distributed and dynamic environments [2]. Trust relationships refer to entities' beliefs that others will provide services as needed [11]. In most cases, entities are inclined to choose partners from the ones in which their beliefs are relatively strong. Entities weaken their beliefs if the found partners cannot provide services. Otherwise, they strengthen their beliefs. They dynamically change the strength values of their beliefs with more and more such experiences. Thus, they can quickly identify partners at any time.

Many studies have been done in applying trust relationships to find partners. Golbeck [2] argues that trust relationships can be used to identify who can produce trustworthy online information. Sen et al. [1] compare three schemes of computing trust relationships in reducing the cost of finding partners from a small group of entities. Specially, they introduce a nearest neighbor algorithm as a scheme through recalling a fixed number of latest experiences of finding partners. To get rid of malicious entities from the group of potential partners, Kamvar et al. [6] build a matrix of weighted trust relationships through aggregating and normalizing based on a history of uploading files. They compute the left principal eigenvector of this matrix as their metric. More related work can be found in [5, 12, 13]. In such work, entities either measure the probabilities that others have successfully provided services before, or ask past partners for their beliefs about third-party ones.

However, it remains to clarify why and how trust relationships can make entities efficiently find partners in dynamic, distributed, real-world networks. This paper aims to solve this problem through proposing a dynamic trust network based on the bottom-up methodology of Autonomy-Oriented Computing (AOC). AOC can help us characterize how network dynamics emerge from local interactions through utilizing the ideas of autonomy and self-organization [7, 9, 8]. Here, network dynamics refers to the dynamics of network structure, network performances in providing services and so forth. Zhang et al. [14] have successfully characterized such dynamics of a distributed network, in which entities work together to provide various

*Prof. Jiming Liu is her corresponding supervisor

services. Their work shows the feasibility and rationality for the work in this paper.

In this study, the autonomy is embodied for entities (nodes) to find partners while a process of self-organization is realized to update trust relationships (links). Entities decide to activate which behaviors for partner search or selection through measuring their abilities and beliefs. They also immediately update their beliefs, once they receive the feedback from their latest partners. Specially, a positive feedback mechanism is emphasized, i.e., the strong beliefs become stronger while the weak become weaker. Experimental results show that our network quickly converges to be scale-free. With such a structure, entities quickly find partners for any request. These results explain why dynamic trust relationships are helpful for entities to find partners.

The reminder of this paper is organized as follows: Section 2 gives a detailed problem statement. Section 3 formulates a network on the basis of AOC. The network contains autonomous entities as nodes and self-organized trust relationships as links. In Section 4, we observe the structural emergence of this network and the dynamics of trust relationships. The efficiency in finding partners is also measured. Finally, the paper is concluded in Section 5.

2 Problem Statement

As mentioned above, the goal of this study is to examine the effects of dynamic trust relationships on finding partners in open, dynamic, and distributed networks, such as the Internet. It requires us to answer the following questions:

- How do entities update their trust relationships? What will they do if they never interact with their newly-found partners before? Will entities memorize the information about all of their past partners?
- What are the mechanisms for entities search and select partners? How do entities refine partners based on updated beliefs? How do entities identify their partners from their refinement results?
- Which parameters can we introduce to characterize the change of trust relationships, besides the variation of their strength? Is it enough to count the number of entities' trustees, associated by their beliefs?
- What is the efficiency in finding partners based on dynamically-changing trust relationships?

Previous studies have confirmed two preliminary phenomena. One is that entities prefer to interact with their trustees rather than strangers. The other is that entities strengthen their beliefs about another entity once it accomplishes a service request and vice versa. Based on the two phenomena, we attempt to answer the above questions in the following scenario.

Here, we assume that there exists a virtual network, which contains trust relationships as links and entities as nodes. In this network, entities are assigned with fixed abilities of providing certain services. Each of them can find a partner to satisfy a whole service request, which cannot be finished by itself. The request has a cost limit, i.e., the maximum number of times for finding partners. The partner will honestly inform whether it can finish this request. Then the finder can decide to strength or weaken its belief about this partner. If this partner cannot finish, it will transfer the request to the third entity, which is discovered as its partner. In other words, this request propagates among entities until it is accomplished or the number of finding times reaches the given cost limit.

Accordingly, the above questions can be translated into the following tasks:

1. Modeling a virtual network, in which distributed entities can find partners and update their beliefs.
2. Measuring structural characteristics at different moments to display the dynamics of the links among entities. The characteristics include network indegree (outdegree), the clustering coefficient and the harmonic mean of average path length (APL^{-1}).
3. Examining the efficiency in finding partners with two parameters, i.e.,
    - Accomplishment ratio: the ratio of the number of accomplished requests to the total number of requests;
    - Average propagation step: the average times that entities find partners for finishing requests.

3 Modeling a Dynamic Trust Network

3.1 Basic ideas

In this section, we will model a virtual network of distributed entities, i.e., a dynamic trust network. Then, we can characterize the structural dynamics of this network, as a reflection of dynamic trust relationships.

To meet the distributed, dynamic needs, we will define the network based on the methodology of AOC. AOC aims to characterize complex networks (e.g., large-scale, dynamic and distributed networks), through deploying a set of autonomy-oriented entities and defining their behavioral rules of self-organization [7, 10]. These entities activate different behaviors to finish a certain task or else, according to their different states, e.g., the number of entities they interacted before. Moreover, entities self-organize their coupling

relationships based on their received feedbacks, such as the state of other entities.

In this work, entities continually find partners in an autonomous manner, as well as self-organize trust relationships based on the feedback from partners.

For finding partners, entities activate one of well-defined search or selection behaviors in a probabilistic manner. The probability of activating each behavior is adaptive to new requests and entities' states (e.g., their beliefs).

A process of self-organization is realized to update trust relationships. At first, there is no trust relationship in the network. After a period of continually finding partners, entities maintain several trust relationships. The realization of this process lies in two aspects. On one side, entities' behaviors are defined to be exploratory or even stochastic. They may find strangers as their partners and create their beliefs about the newly-found partners. However, with positive feedback mechanisms, the newly-created beliefs may be weakened or even eliminated later. Therefore, only a part of generated relationships can remain. The remaining relationships can help entities effectively find partners to finish new requests. So, any request can be accomplished with a higher probability.

The network, entities and service requests are defined as follows:

Definition 1 *Let* $G^{'}(t)=< E, L^{'}(t) >$ *denote a dynamic network of trust-based entities on the basis of AOC. For brevity, we call it a dynamic trust network.* $E = \{e_1, \cdots, e_{N_e}\}$ *denotes the set of entities and* N_e *is the number of entities in the network.* $L^{'}(t) = \{l^{'}_{ij}(t)|e_i, e_j \in E\}$ *is the set of trust relationships at time* t.

Definition 2 *Let* $e_i =< ID, ability, rules >$ *denote an entity where* ID *is its identifier, i.e.,* $e_i.ID = i, i \in [1, N_e]$. *It can provide some services with different performances, i.e.,* $e_i.ability =< cw_1, \cdots, cw_{N_{service}} >$, *where* cw_j *denotes the performance on the* j*th type of service* t_j. $N_{service}$ *is the number of service types. The rules define when and how entities activate their behaviors.*

Definition 3 *A request can be formulated as* $r_m=< rw_1, \cdots, rw_{N_{service}}, T_{max} >$ *where* rw_j *denotes the performance requirement of the type of service* t_j. T_{max} *is the maximum number of times that entities are allowed to find partners.*

The two ideas from AOC are illustrated through describing the process of finding partners for finishing a request, which is specified in Section 3.5. Once a new request r_m is submitted to an entity e_i, this entity will perform the following:

1. **Evaluating**. Firstly, the entity will determine whether it can solely accomplish the new request by means of matching its ability with this request using a cosine-based similarity function $e_i.simRE(r_m)$. The request can be considered as accomplished once the value of $e_i.simRE(r_m)$ is larger than a threshold. The evaluation functions are specified in Section 3.4.

2. **Partner Search and Selection**. If the request cannot be accomplished and its T_{max} is not reached, the entity e_i will start finding partners. Firstly, it will search some candidates by means of activating one of search behaviors in a probabilistic manner. Then, it will select a candidate as its partner through activating one of selection behaviors also probabilistically. The probabilities are adaptive to the request and the states of the entity, e.g., its trust relationships. So, trust relationships will explicitly affect entities' partner finding. The detailed behaviors are given in Section 3.2.

3. **Updating**. The entity e_i will change its states once a partner is found. Firstly, it will deliver the whole request to its partner. Then, the partner e_j will honestly feedback its evaluating results $e_j.simRE(r_m)$. Finally, this entity will generate the trust relationship $l^{'}_{ij}$ if $l^{'}_{ij}(t)$ is not in the network. Otherwise, it will strengthen the relationship if the ability of this partner is relatively closer to this request, or weaken the trust relationship if not. The mechanisms of trust relationships are specified in Section 3.3.

3.2 The Local Autonomy of Entities

Three search behaviors and two selection behaviors are defined in this section. They will be probabilistically activated to realize the autonomy of entities. The probability of activating each behavior is determined with the degree of similarity between requests and entities' abilities, i.e., $e_i.simRE(r_m)$. When the degree is high, entities' neighbors will be discovered as partners with a high probability. Stochastic behaviors are given for entities to avoid being trapped in local-optima and for newcomers to join this network.

The detailed operations of search and selection behaviors are specified in below:

- **Neighbor-based search** $neighborSearch()$. An entity e_i will find some neighbors, which are not involved in the current request r_m, with the probability of $e_i.simRE(r_m)$. The probability will be adaptive to requests accordingly. Neighbors refer to its trustees. i.e., there are trust relationships from this entity to them.

- **Recommendation-based search** $recommendSear-ch()$. An entity e_i will discover candidates within a distance $maxD$ around itself with a probability of

$1 - e_i.simRE(r_m)$ when the receiving request is relatively near the entity's ability. That is, the entity will enlarge its search area from its neighbors to the area within a given distance. This behavior is inspired by two observations: 1) besides direct experiences, indirect experiences are also important for entities to make decisions; 2) communities have been discovered in many real-world networks, i.e., agents with similar interests cluster together. Here, the request far beyond the ability of an entity is likely out of its neighbors' abilities.

- **Random search** $randomSearch()$. An entity e_i will search in the whole network when it is a newcomer or its neighbors are all involved in the current request.
- **Trust-based select** $trustSelect()$. The entity e_i will select an entity e_j with the maximal degree of trust, $e_i.trust(e_j, t)$, from its candidates as its partner with a probability of $e_i.simRE(r_m)$. Entities are supposed to be more confident in selecting partners for requests which are relatively closer to its abilities.
- **Random select** $randomSelect()$. The entity e_i will stochastically select an entity from the candidate set $e_i.cand(r_m)$ as its partner with a probability of $1 - e_i.simRE(r_m)$.

3.3 The Mechanisms of Trust Relationships

We have introduced how trust relationships are applied in activating entities' behaviors for finding partners in the previous section. Now, we will introduce how they are changed based on the feedback from partners. Trust relationships are defined as follows:

Definition 4 *A link* l'_{ij}=$\{< e_i, e_j, e_i.succ(e_j,t), e_i.fail(e_j,t), e_i.latestTime(e_j), e_i.trust(e_j,t) >| 1 \leq i,j \leq N_e\}$ *reflects a trust relationship from* e_i *to* e_j *where*

- $e_i.succ(e_j,t)$ *and* $e_i.fail(e_j,t)$ *denote the numbers of successful and failed interactions, respectively;*
- $e_i.latestTime(e_j)$ *is the time of their latest interaction;*
- $e_i.trust(e_j,t)$ *is the degree of trust at time t, quantifying the belief that* e_j *will accomplish a new request from* e_i*. It is a numeric value in* $[\varepsilon, 1.0]$*, where the threshold* ε *will be described below.*

Once an entity e_i receives the feedback of its partner $e_j.simRE(r_m)$, a new relationship $l_{ij}(t)$ will be generated if it is not in the network. Otherwise, the existing relationship will be updated. The parameters of this relationship will be assigned as introduced below.

Firstly, the time of their latest interaction is set as the current time. Then, e_i will evaluate whether the partner is more suitable for the current request with Eq. 1 where λ is a threshold, $\lambda \in (0, 1)$.

$$e_i.QoI(e_j, r_m) = \begin{cases} true & \frac{e_j.simRE(r_m)}{e_i.simRE(r_m)} > (1+\lambda); \\ false & otherwise \end{cases} \quad (1)$$

Other parameters will be assigned according to the result of $e_i.QoI(e_j)$. When the partner is more suitable for this request, i.e., $e_i.QoI(e_j, r_m) = true$, this interaction is regarded as successful and the number of their successful interactions increases. Otherwise, the number of failed interactions increases. Then, the degree of trust $e_i.trust(e_j,t)$ can be set with updated $e_i.succ(e_j,t)$ and $e_i.fail(e_j,t)$ as follows:

$$e_i.trust(e_j,t) = \frac{e_i.succ(e_j,t)}{e_i.succ(e_j,t) + e_i.fail(e_j,t)} \quad (2)$$

Trust relationships may be removed since the degree of trust is supposed to decay over time as follows:

$$\begin{aligned} e_i.trust(e_j,t) = e_i.trust(e_j, e_i.latestTime(e_j)) \\ - \eta * (t - e_i.latestTime(e_j)) \end{aligned} \quad (3)$$

where η is the decay factor, $\eta \in (0.0, 1.0)$. Once the degree of trust is less than a small negative numeric value ε, i.e., $e_i.trust(e_j,t) < \varepsilon$, the neighbor is interpreted as being no longer able to accomplish new requests from this entity, and the corresponding relationship will be eliminated. The threshold ε is empirically set, $\varepsilon \in (-0.05, 0)$.

In addition, entities will derive new trust relationships from the relationships already in the network. If the shortest distance from entity e_i to entity e_j is larger than a given threshold $maxD$, i.e., $e_i.shortestDis(e_j,t) > maxD$, e_i has no idea about the ability of e_j and the degree of trust $e_i.trust(e_j,t)$ is set as 0. Otherwise, the following functions can be used to compute the degree of trust for the derived relationships:

$$\begin{aligned} e_i.trust(e_j,t) &= \frac{\sum_{k=1}^{n_{ij}(t)} e_i^k.trustPath(e_j,t)}{n_{ij}(t)} \\ e_i^k.trustPath(e_j,t) &= \sum_{l'_{m_j}(t) \in L'(t)} (e_i^k.trustPath(e_m,t) \\ &\quad \times e_m.trust(e_j,t)) \end{aligned} \quad (4)$$

where $n_{ij}(t)$ is the number of the shortest paths from e_i to e_j and the function $e_i^k.trustPath(e_j,t)$ denotes the degree of trust derived from the kth shortest path. If e_j is a neighbor of e_i, $e_i^k.trustPath(e_j,t)$ will be computed with Eq. 3.

3.4 Evaluation Functions

We define two functions, $e_i.simRE(r_m)$ and $e_i.match(r_m)$, for entities to evaluate whether they can accomplish received requests in this section. A cosine-based function is given to compute the degree of similarity between request r_m and the ability of entity e_i, i.e.,

$$e_i.simRE(r_m) = \frac{\sum_{k=1}^{N_{service}} (r_m.rw_k * e_i.cw_k)}{\sqrt[2]{\sum_{k=1}^{N_{service}} r_m.rw_k^2 * \sum_{k=1}^{N_{service}} e_i.cw_k^2}} \quad (5)$$

When the value of $e_i.simRE(r_m)$ is larger than a threshold δ, $\delta \in (0, 1)$, the request r_m is assumed to be accomplished by the entity e_i, i.e.,

$$e_i.match(r_m) = \begin{cases} true & simRE(r_m) > \delta; \\ false & otherwise \end{cases} \quad (6)$$

3.5 The Algorithm

The trust network will evolve through continually finding partners based on the above-defined behaviors and trust relationships. Algorithm 1 describes how entities find partners once a new request is stochastically submitted. $search()$ and $select()$ represent entities activate a behavior of search and selection, respectively. $updateState()$ denotes that entities update their trust relationships. As defined before, $G'(t)$ represents the trust network at time t, i.e., t requests have been submitted to the network. If no request has ever been submitted, the network $G'(t)$ only contains N_e independent entities, i.e., $G'(0) =< \{e_1, \cdots, e_{N_e}\}, \emptyset >$.

Algorithm 1: The Autonomy-Oriented Partner Finding

Input: A new request r_m, the dynamic trust network $G^{'}(t)$
Output: The evolved network $G^{'}(t+1)$
begin
  `// initialization phase`
  stochastically select an entity e_i as the receiver of new request r_m;
  $current\_entity \leftarrow e_i$; $next\_entity \leftarrow null$;
  $flag \leftarrow true$;
  `// self-organized computing phase`
  while $r_m.T_{max} > 0$ **do**
    $matchResult \leftarrow current\_entity.match(r_m)$ based on Eq. 6;
    if $matchResult \neq$ *true* **then**
      $current\_entity.search()$;
      $next\_entity \leftarrow current\_entity.select()$;
      `// update based on feedback`
      $flag \leftarrow current\_entity.QoI(next\_entity, r_m)$ based on Eq. 1;
      $current\_entity.updateState(flag)$ based on Eq. 2;
      $current\_entity \leftarrow next\_entity$;
      $r_m.T_{max} = r_m.T_{max} - 1$;
    else
      $r_m.T_{max} = 0$;
    end
  end
end

4 Experiments

In the above, we have presented a dynamic trust network, which contains autonomous entities as nodes and self-organized trust relationships as links. This section gives some experiments toward the following objectives:

1. to characterize the dynamics of trust relationships by means of measuring network structural characteristics;
2. to measure the efficiency of the network in finding partners to accomplish requests.

4.1 Experimental Setting

Experimental parameters are listed in Table 1. The simulated network contains 1000 entities and can provide 5 types of services. Entities' abilities on providing each type of service are supposed to follow a power-law distribution and the power falls in the range of $[1, 2]$. That is, most entities provide certain type of service with a low performance, e.g., a low quality of service. A cycle denotes the time entities spent on finishing a request and updating corresponding trust relationships among them. The performance requirement and T_{max} of all requests are stochastically initialized. For each request, its requirement on each type of service is set to be in the field of $[0.0, 1.0]$ while its T_{max} is assigned with a random integer value in $(0, 999]$. The number of entities determines the upper limit of T_{max} since each entity is allowed to be found once for the same request. For fairness, we submit the same set of 100 requests after each cycle and test the efficiency of entities in finishing such requests.

4.2 The Dynamics of Trust Relationships

We examine the dynamics of trust relationships by measuring the structure characteristics, including the network

Table 1: The experimental parameters

$N_{service} = 5$	$N_{cycles} = 1500$	$N_{testRequest} = 100$
$N_e = 1000$	$\lambda = 0.2$	$maxD = 4$
$\delta = 0.95$	$\eta = 0.0003$	$\varepsilon = -0.02$

indegree (outdegree), the clustering coefficient, the harmonic mean of average path length APL^{-1} and the distributions of entities' indegree and outdegree. The results are shown in Fig. 1 and Fig. 2.

In Fig. 1, the values of three characteristics fluctuate relatively slightly after changing rapidly in the first 250 cycles. That means that entities dramatically generate and remove trust relationships at the beginning. Then, they find and maintain the relationships to their stable partners for any request later.

In Fig. 2, entities' indegree or outdegree approximately follow power-law distribution in the 100th, 500th, 1000th, 1500th cycles. Specially, we have observed that the phenomenon of a power-law distribution appears firstly at the 16th cycle. The power dynamically changes in the following cycles, as a result of more and more entities involve in finishing requests and then update their trust relationships. The result is quite interesting. It can be inferred that entities can more efficiently find their partners in such a structure, as to be discussed in the following section.

4.3 The Dynamics of Efficiency

In this section, we will examine the efficiency of our network in finding partners to finish the fixed 100 requests in different cycles. The measurements include:

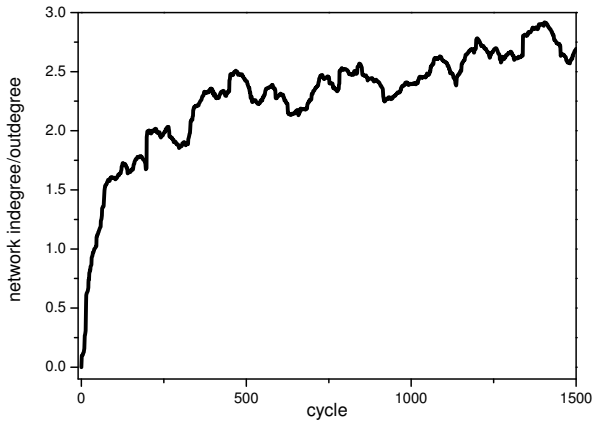
- the accomplishment ratio $accompRatio$: the proportion of the number of accomplished requests to 100;
- the average steps $avgStep$: the average times entities find partners for finishing these requests;
- the standard deviation of propagation steps $stDevOfStep$.

As to be described later, the accomplishment ratio gradually increases, while the average steps and the standard deviation decrease. The efficiency gets enhanced with the change of trust relationships.

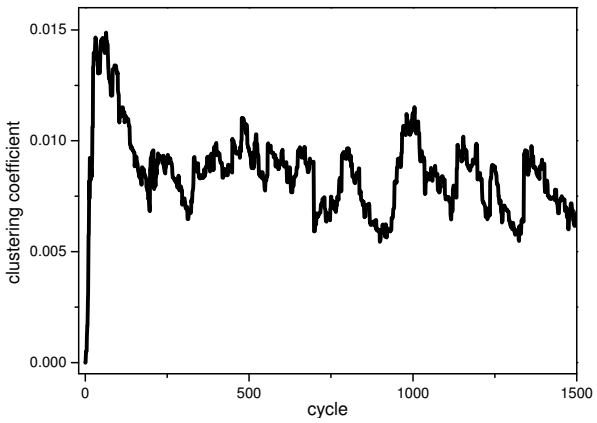
Table 2: The efficiency at different cycles

cycle	$accompRatio$	$avgStep$	$stDevOfStep$
0	0.937	52.242	61.603
1500	0.958	43.702	49.838

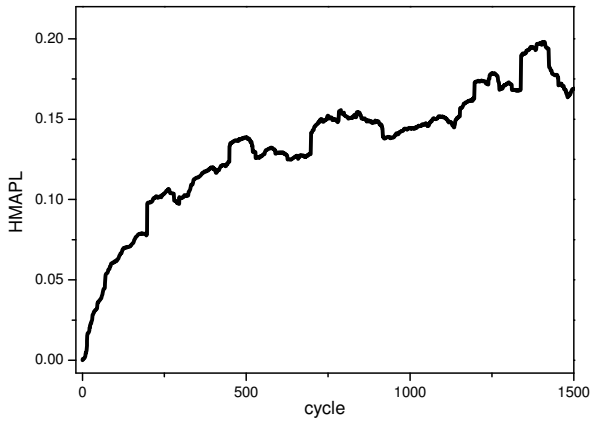
Fig.3 presents the dynamics of efficiency. Each point in curves is computed over 10 cycles. Entities may only



(a) the network indegree (outdegree)

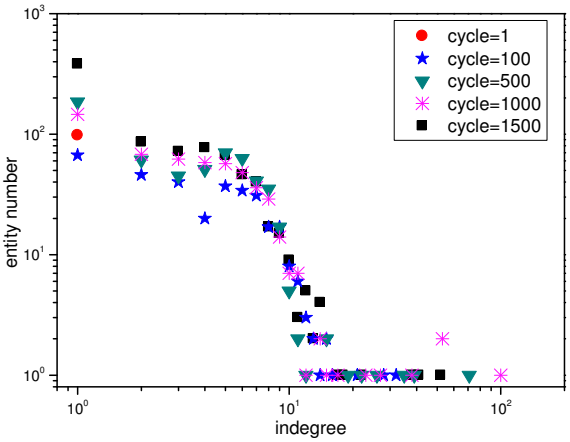


(b) the clustering coefficient

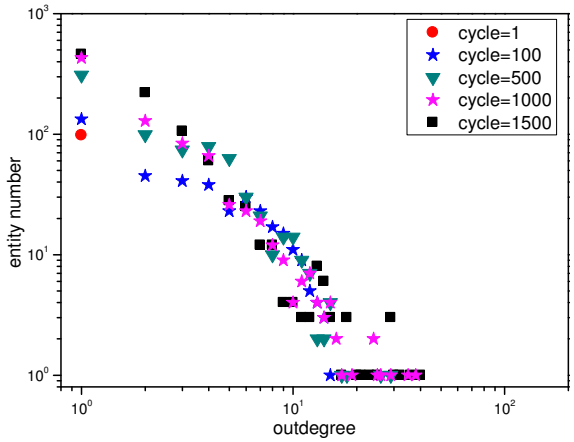


(c) the APL^{-1}

Figure 1: The dynamics of trust relationships with the measurement of (a) network indegree (outdegree), (b) clustering coefficient and (c) the harmonic mean of average path length.

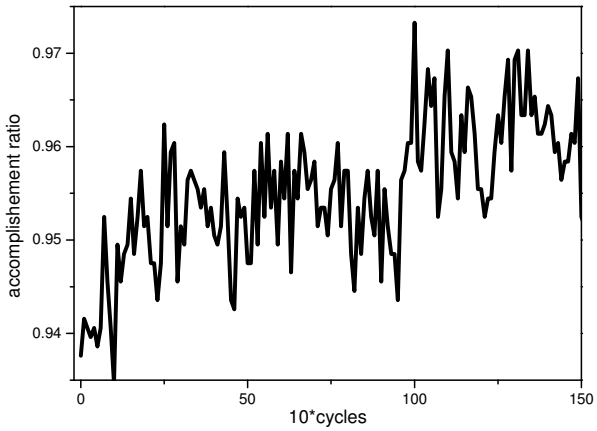


(a) the distribution of indegree

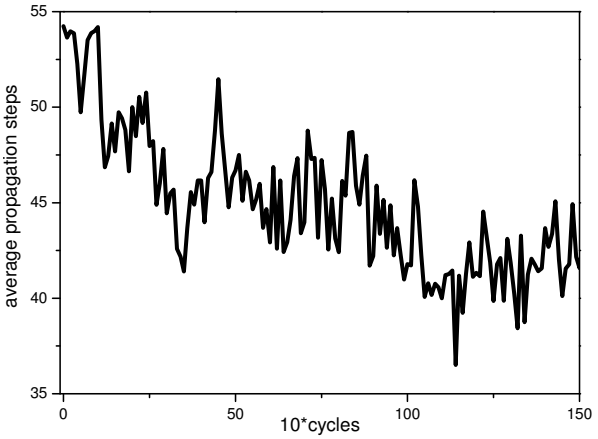


(b) the distribution of outdegree

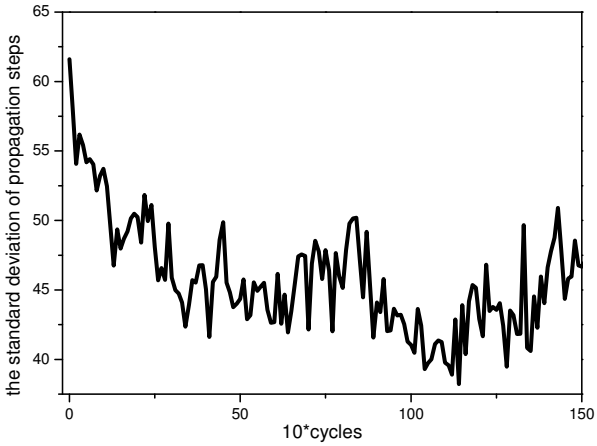
Figure 2: The distributions of (a) entities' indegree and (b) outdegree at different cycles. Entities' indegree and outdegree approximately follow power-law distributions since the 16th cycle.



(a) the accomplishment ratio



(b) the average steps



(c) the standard deviation of propagation steps

Figure 3: The dynamics of the efficiency with the three measurements: (a) the accomplishment ratio of 100 requests, (b) the average number of the times entities find partners for finishing the requests, (c) the standard deviation of the times spent on finding partners.

update 2 or 3 relationships in a cycle of updating, which hardly affects the efficiency. We can find that the upward or downward tendency in each sub-graph is quite clear. Table 2 gives the efficiency at the beginning cycle and ending cycle. The average steps decrease about 17.653%. It potently shows that the change of trust relationships leads to the high efficiency of finding partners.

5 Concluding Remarks

In this study, we are interested in characterizing dynamically-changing trust relationships in real-world applications. We attempt to understand why trust relationships can enable distributed entities quickly find partners to provide their requested services.

Specifically, we have proposed a dynamic trust network by means of utilizing the ideas of autonomy and self-organization from AOC. These ideas can help us understand how network-level phenomena emerge from entities' local behaviors. In this study, the notion of autonomy means that distributed entities (nodes) activate different behaviors based on their abilities and trust relationships. Self-organization refers to the process that entities change their relationships (links) with positive feedback mechanisms, according to the feedback from their partners.

Experimental results have shown that the network quickly converges to be scale-free. In other words, parts of dynamically-generated trust relationships remain relatively stable while others are eliminated. That is, the process of self-organization makes trust relationships emerge quickly. Meanwhile, the accomplishment ratio gradually rises while the average steps and the standard deviation of propagation steps gradually decrease. It can be inferred that, the convergence process of this network is accomplished with the enhancement of its efficiency in successfully finding partners.

References

[1] L. Erete, E. Ferguson, and S. Sen. Learning task-specific trust decisions. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multi-Agent Systems*, 1477–1480, 2008.

[2] J. Golbeck. Weaving a web of trust. *Science*, 321(5896):1640–1641, 2008.

[3] M. Greeger. CTO roundtable: Cloud computing. *Communications of the ACM*, 52(8):50–56, 2009.

[4] B. Hayes. Cloud computing. *Communications of the ACM*, 51(7):9–11, 2008.

[5] A. Jøsang, R. Ismail, and C. Boyd. A survey of trust and reputation systems for online service provision. *Decision Support Systems*, 43:618–644, 2007.

[6] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The eigentrust algorithm for reputation management in P2P networks. In *Proceedings of the 12th International World Wide Web Conference*, 640–651, 2003.

[7] J. Liu. Autonomy-oriented computing (AOC): The nature and implications of a paradigm for self-organized computing (keynote talk). In *Proceedings of the 4th International Conference on Natural Computation and the 5th International Conference on Fuzzy Systems and Knowledge Discovery*, 3–11, 2008.

[8] J. Liu, X. Jin, and K. C. Tsui. Autonomy-oriented computing (AOC): Formulating computational systems with autonomous components. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 35(6):879–902, 2005.

[9] J. Liu, X. Jin, and K. C. Tsui. *Autonomy Oriented Computing: From Problem Solving to Complex Systems Modeling*. Springer, 2005.

[10] J. Liu and K. Tsui. Toward nature-inspired computing. *Communications of the ACM*, 49(10):59–64, 2006.

[11] S. P. Marsh. *Formalising Trust as a Computational Concept*. PhD thesis, University of Stirling, 1994.

[12] S. D. Ramchurn, T. D. Huynh, and N. R. Jennings. Trust in multi-agent systems. *The Knowledge Engineering Review*, 19(1):1–25, 2004.

[13] J. Sabater and C. Sierra. Review on computational trust and reputation models. *Artificial Intelligence Review*, 24:33–60, 2005.

[14] S. Zhang and J. Liu. Autonomy-oriented social networks modeling: Discovering the dynamics of emergent structure and performance. *International Journal of Pattern Recognition and Artificial Intelligence*, 21(4):611–638, 2007.